Trabalho Final

CI1030 - Ciência de Dados para Segurança

Alunos: Alessandro Luz
Diogo Bortolini

Prof. Doutor: André Ricardo Abed Grégio

2021

DATASET

Intrusion Detection Evaluation Dataset
CICIDS2017 - Disponível em: https://www.unb.ca/cic/datasets/ids-2017.html

Dataset de segurança, gerado com 5 dias com ataques de rede e tráfego benigno.

Utilizado Porção de Ataque WEB (1 dia) - Categorizados
Tráfego BENIGNO, Web Attack – Brute Force, Web Attack – XSS, Web Attack – Sql Injection

Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv
458968 registros
85 colunas

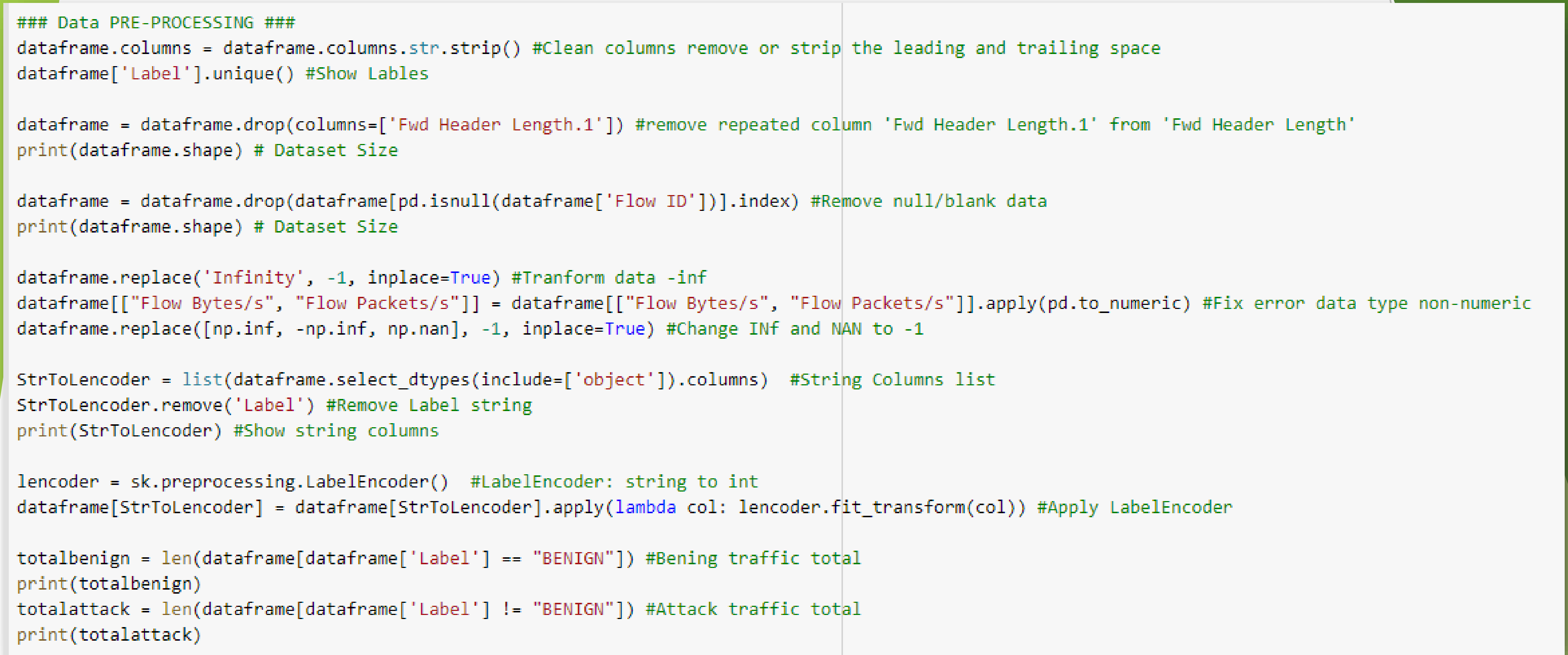
RECURSOS

Github: https://github.com/diogobortolini/CI1030-Ciencia-de-Dados-para-Seguranca

Google Colab

Python com: `pandas – numpy – sklearn – matplotlib`

Algoritmos ML: `kNN – RandomForest – MLP`



```
### Data PRE-PROCESSING ###
dataframe.columns = dataframe.columns.str.strip() #Clean columns remove or strip the leading and trailing space
dataframe['Label'].unique() #Show Lables

dataframe = dataframe.drop(columns=['Fwd Header Length.1']) #remove repeated column 'Fwd Header Length.1' from 'Fwd Header Length'
print(dataframe.shape) # Dataset Size

dataframe = dataframe.drop(dataframe[pd.isnull(dataframe['Flow ID'])].index) #Remove null/blank data
print(dataframe.shape) # Dataset Size

dataframe.replace('Infinity', -1, inplace=True) #Tranform data -inf
dataframe[["Flow Bytes/s", "Flow Packets/s"]] = dataframe[["Flow Bytes/s", "Flow Packets/s"]].apply(pd.to_numeric) #Fix error data type non-numeric
dataframe.replace([np.inf, -np.inf, np.nan], -1, inplace=True) #Change INf and NAN to -1

StrToLencoder = list(dataframe.select_dtypes(include=['object']).columns)  #String Columns list
StrToLencoder.remove('Label') #Remove Label string
print(StrToLencoder) #Show string columns

lencoder = sk.preprocessing.LabelEncoder()  #LabelEncoder: string to int
dataframe[StrToLencoder] = dataframe[StrToLencoder].apply(lambda col: lencoder.fit_transform(col)) #Apply LabelEncoder

totalbenign = len(dataframe[dataframe['Label'] == "BENIGN"]) #Bening traffic total
print(totalbenign)
totalattack = len(dataframe[dataframe['Label'] != "BENIGN"]) #Attack traffic total
print(totalattack)
```

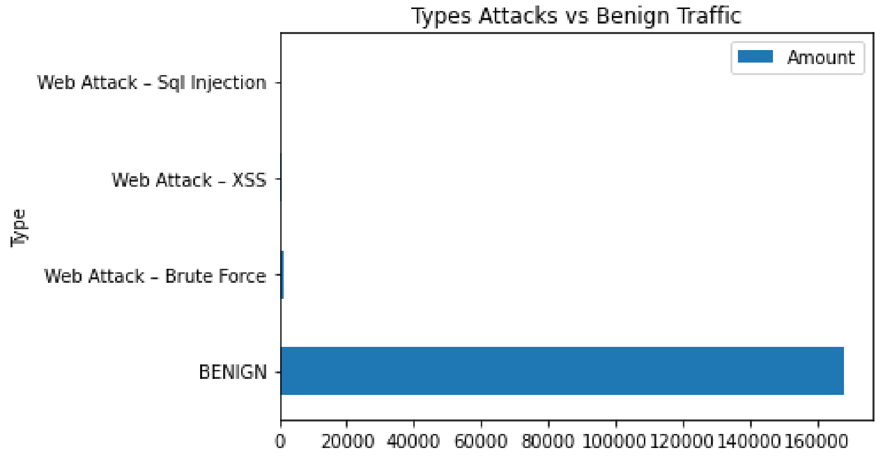
PRÉ-PROCESSAMENTO

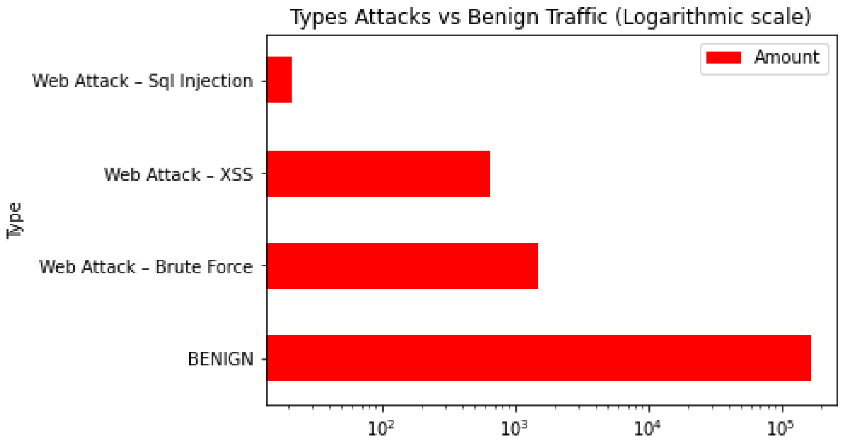
(170366, 84)

Registros Colunas

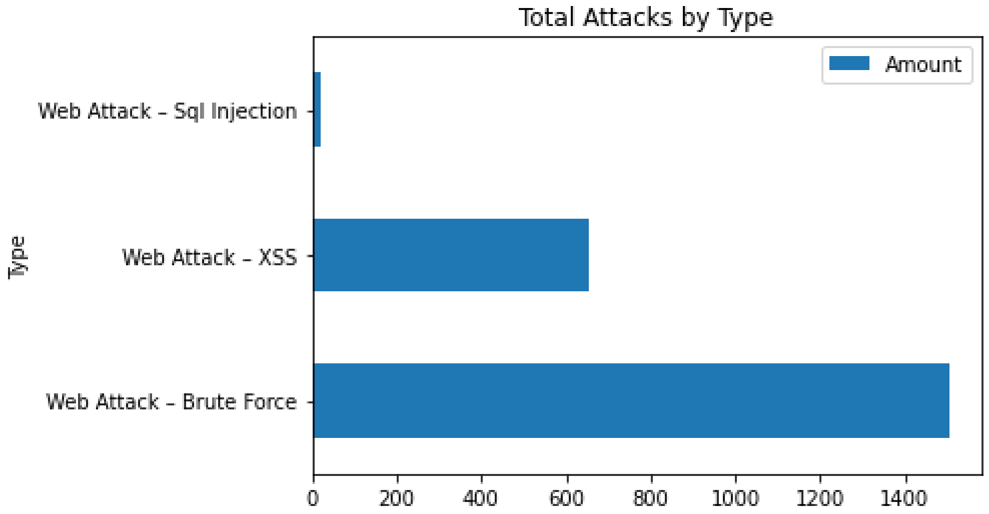
(168186, 2180)

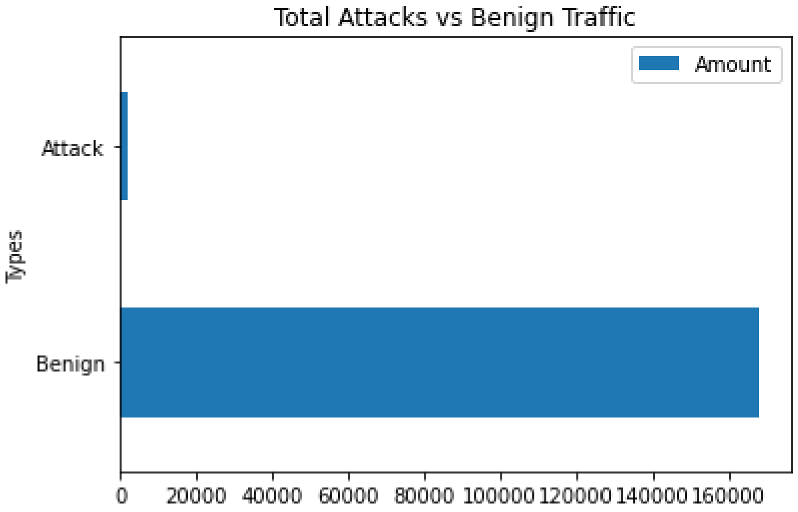
Benigno Ataques


Types Attacks vs Benign Traffic

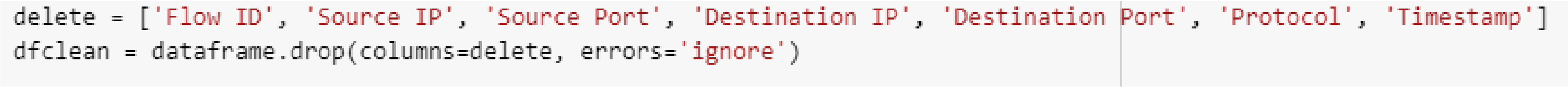

Types Attacks vs Benign Traffic (Logarithmic scale)

Distribuição


Total Attacks by Type

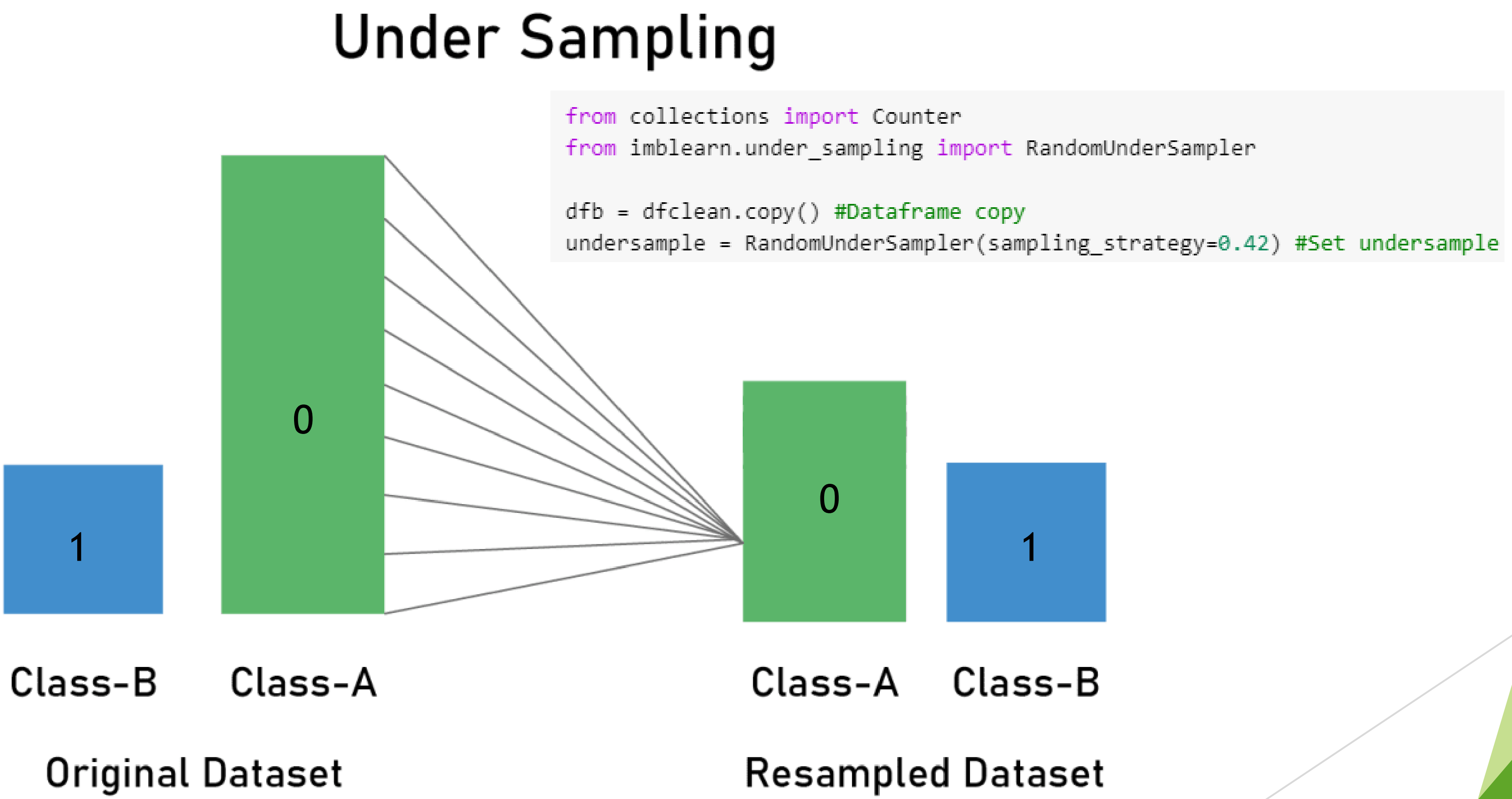

Total Attacks vs Benign Traffic

Exclusão de outros dados

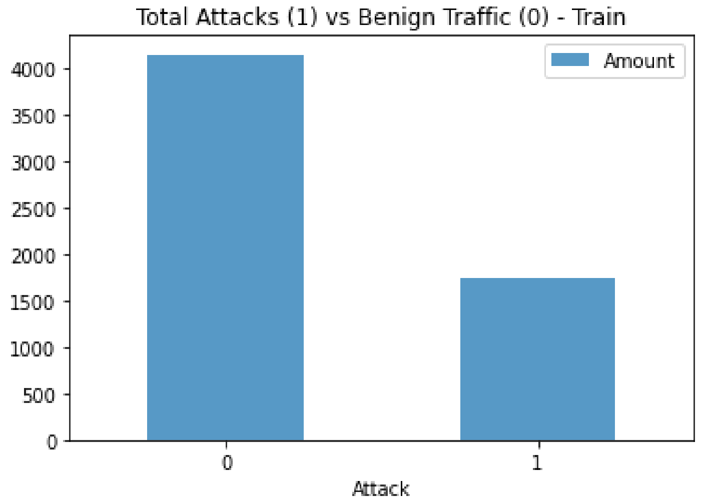


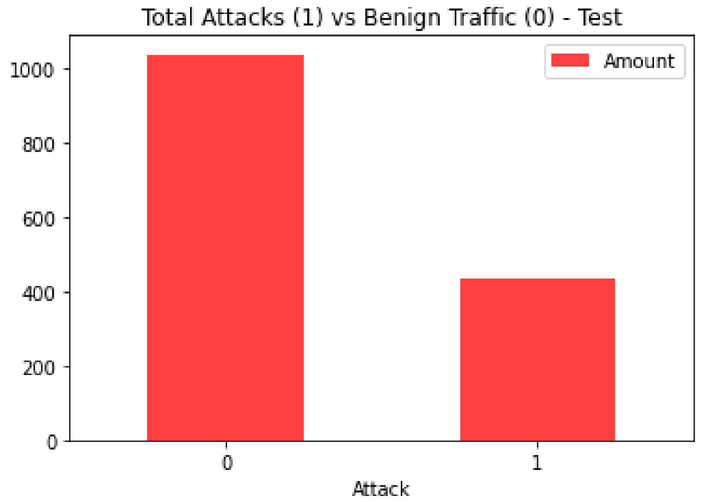
```
delete = ['Flow ID', 'Source IP', 'Source Port', 'Destination IP', 'Destination Port', 'Protocol', 'Timestamp']
dfclean = dataframe.drop(columns=delete, errors='ignore')
```

Balanceamento dos Dados

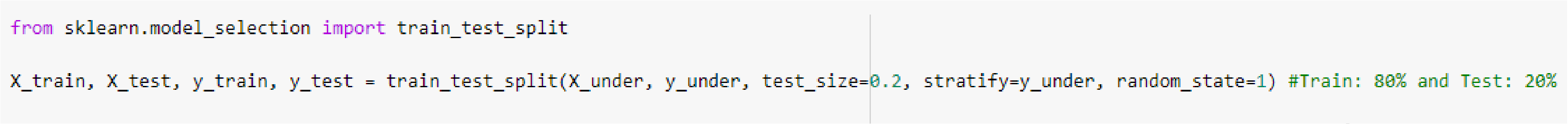


```
from collections import Counter
from imblearn.under_sampling import RandomUnderSampler

dfb = dfclean.copy() #Dataframe copy
undersample = RandomUnderSampler(sampling_strategy=0.42) #Set undersample
```

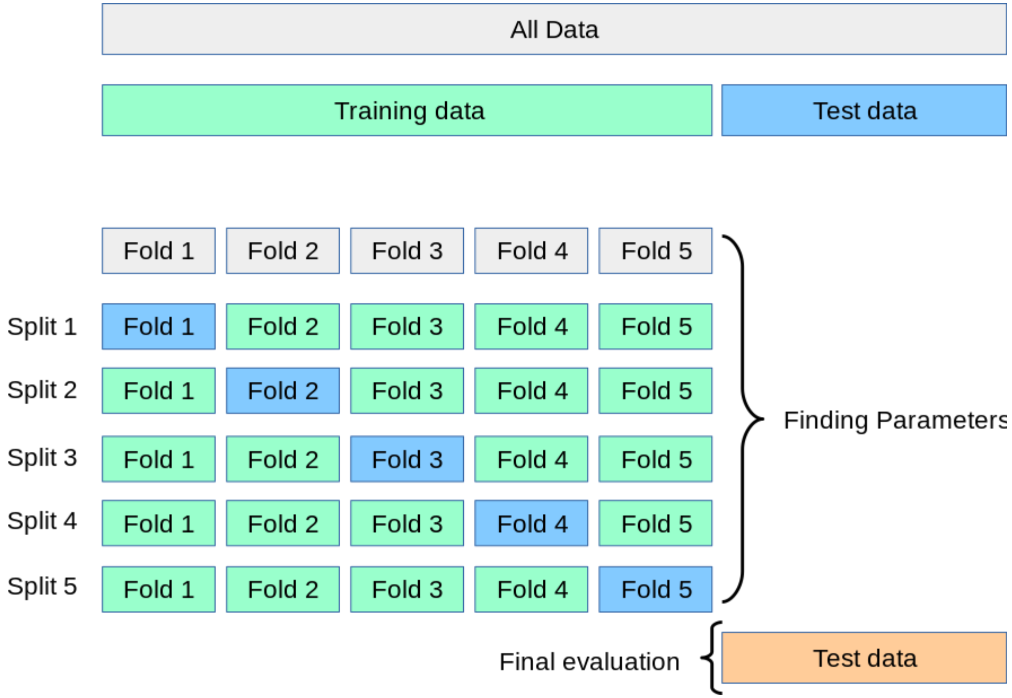




Divisão em Treino e teste



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_under, y_under, test_size=0.2, stratify=y_under, random_state=1) #Train: 80% and Test: 20%
```



All Data

Training data | Test data

Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 1 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 2 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

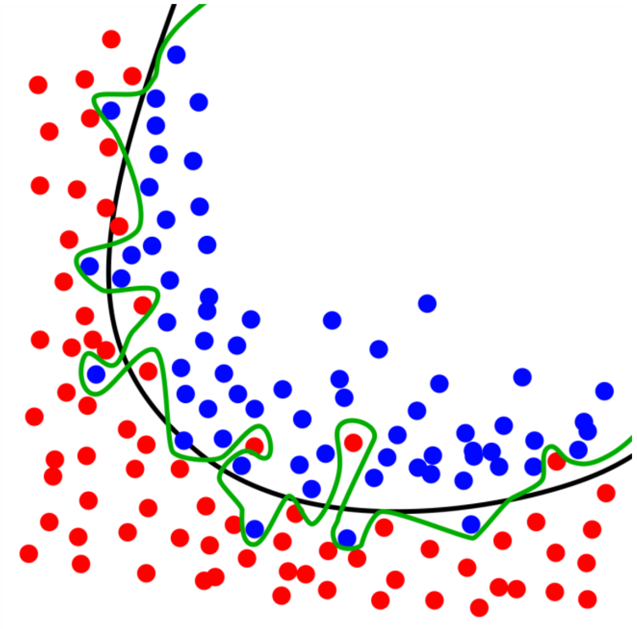
Split 3 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 4 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

Split 5 | Fold 1 | Fold 2 | Fold 3 | Fold 4 | Fold 5

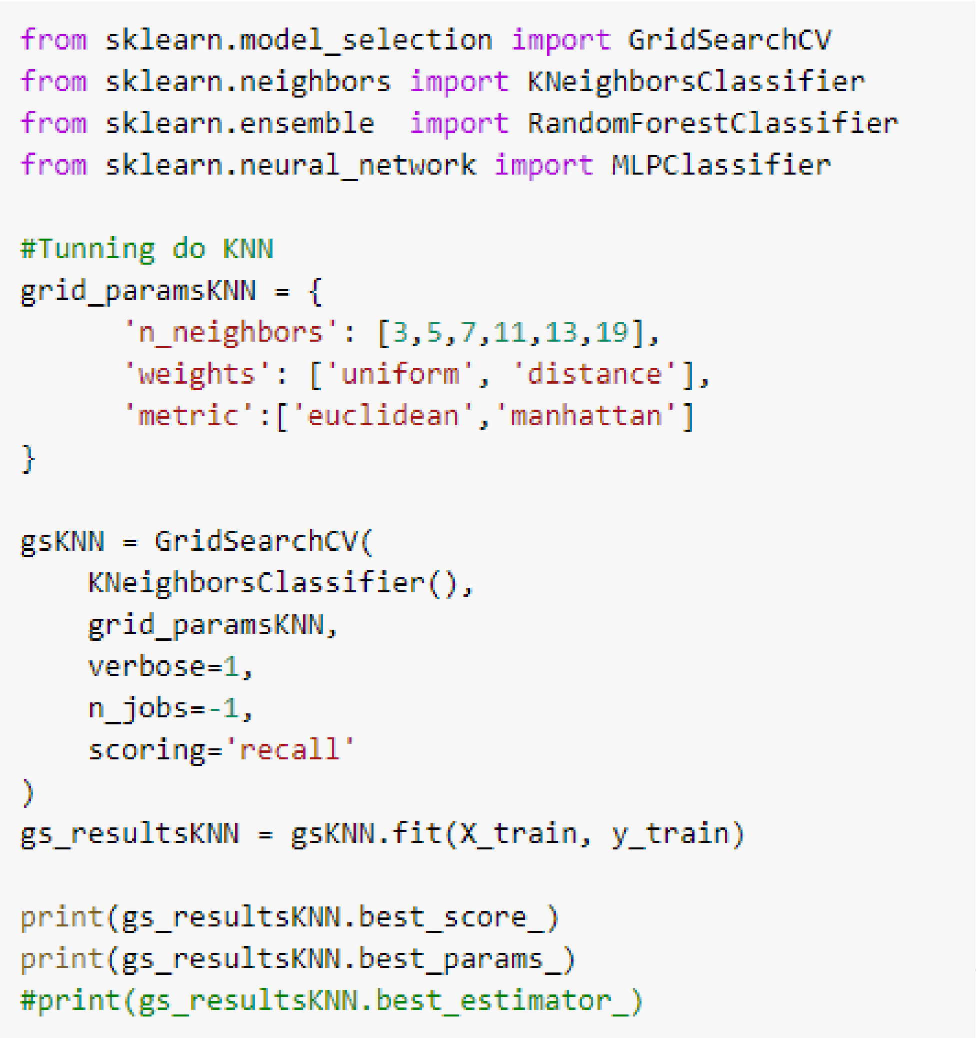
Finding Parameters

Final evaluation | Test data

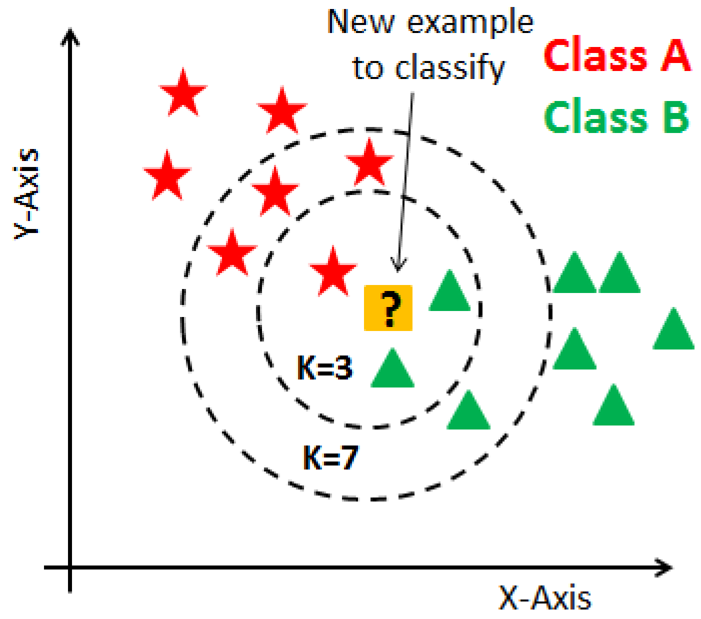


Divisão em Treino e teste

Tunning dos modelos



```
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble  import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

#Tunning do KNN
grid_paramsKNN = {
      'n_neighbors': [3,5,7,11,13,19],
      'weights': ['uniform', 'distance'],
      'metric':['euclidean','manhattan']
}

gsKNN = GridSearchCV(
    KNeighborsClassifier(),
    grid_paramsKNN,
    verbose=1,
    n_jobs=-1,
    scoring='recall'
)
gs_resultsKNN = gsKNN.fit(X_train, y_train)

print(gs_resultsKNN.best_score_)
print(gs_resultsKNN.best_params_)
#print(gs_resultsKNN.best_estimator_)
```

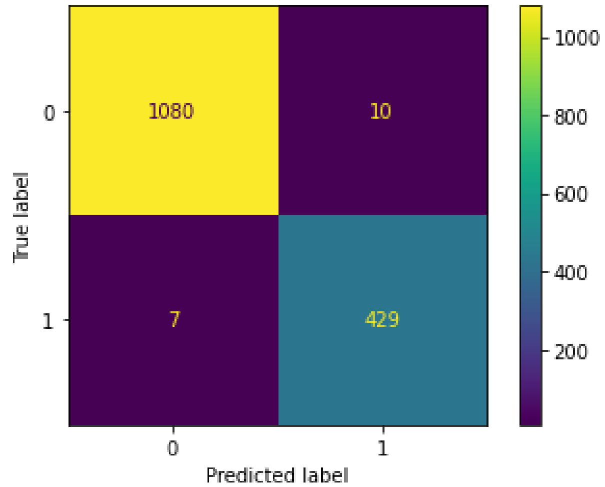


K-NN

K-NN

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1090
           1       0.98      0.98      0.98       436

    accuracy                           0.99      1526
   macro avg       0.99      0.99      0.99      1526
weighted avg       0.99      0.99      0.99      1526

Confusion Matrix:
[[1080   10]
 [   7  429]]
```

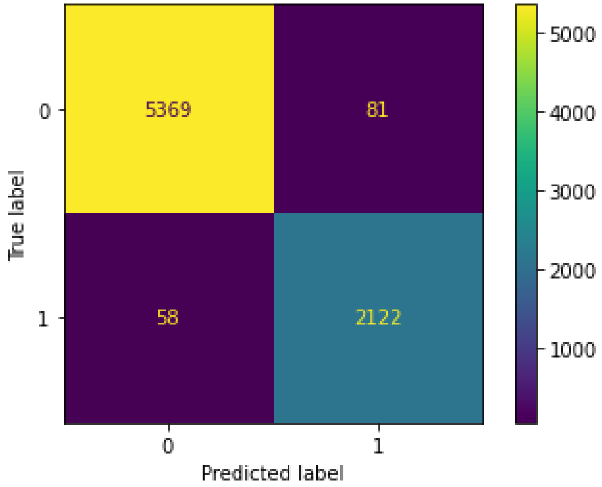
Confusion Matrix KNN



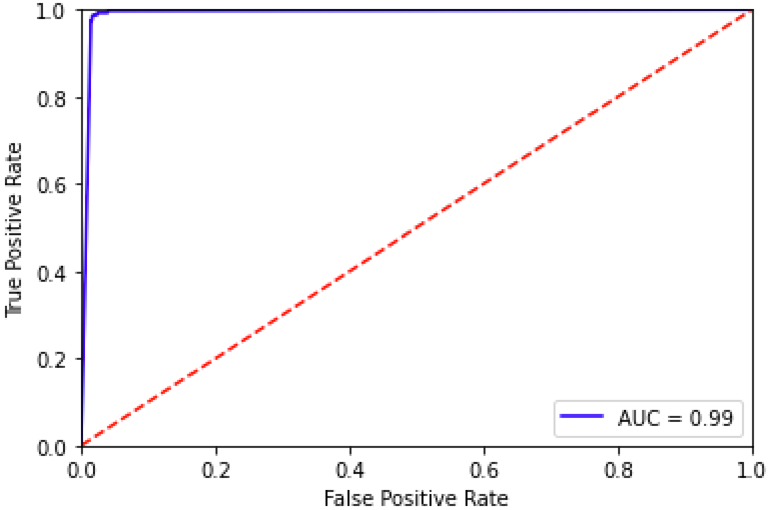
```
Acurácia Média: 0.98 | Desvio: 0.0051
Precisão Média: 0.96 | Desvio: 0.0035
Revocação Média: 0.97 | Desvio: 0.018
F1 Score Média: 0.97 | Desvio: 0.0093
```

```
Confusion Matrix kfold:
[[5369   81]
 [  58 2122]]
```

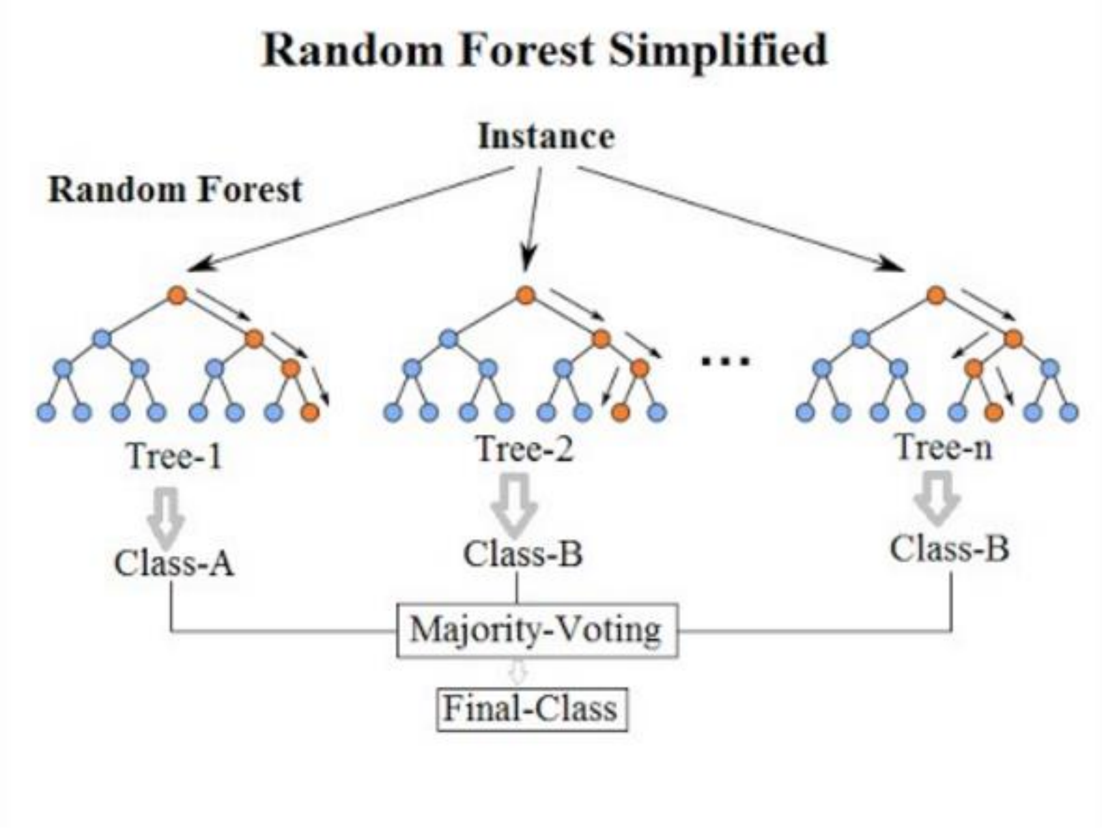
Confusion Matrix KNN kFolds



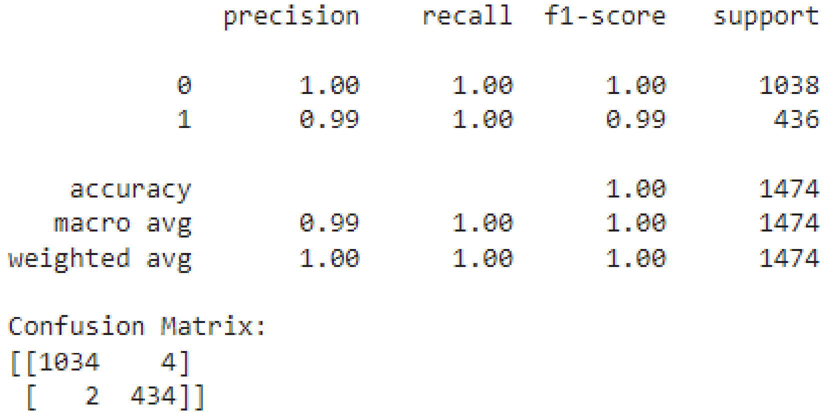
ROC Curve of kNN



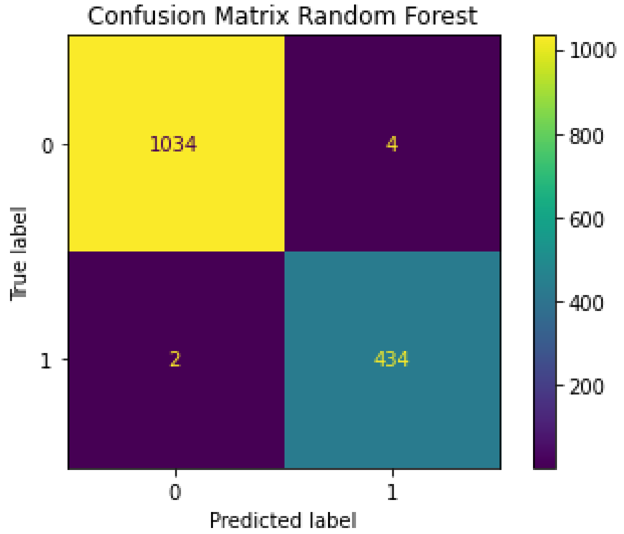
Random Forest

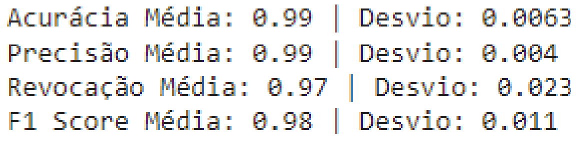


Random Forest

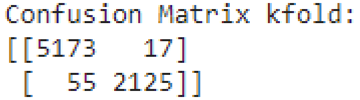


```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1038
           1       0.99      1.00      0.99       436

    accuracy                           1.00      1474
   macro avg       0.99      1.00      1.00      1474
weighted avg       1.00      1.00      1.00      1474

Confusion Matrix:
[[1034    4]
 [   2  434]]
```

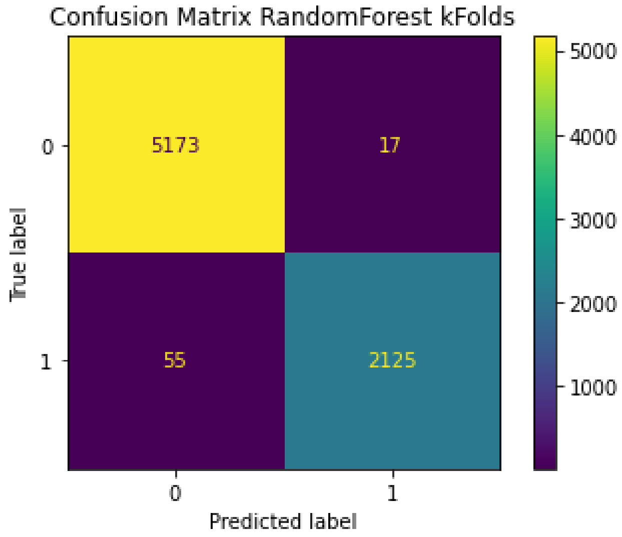


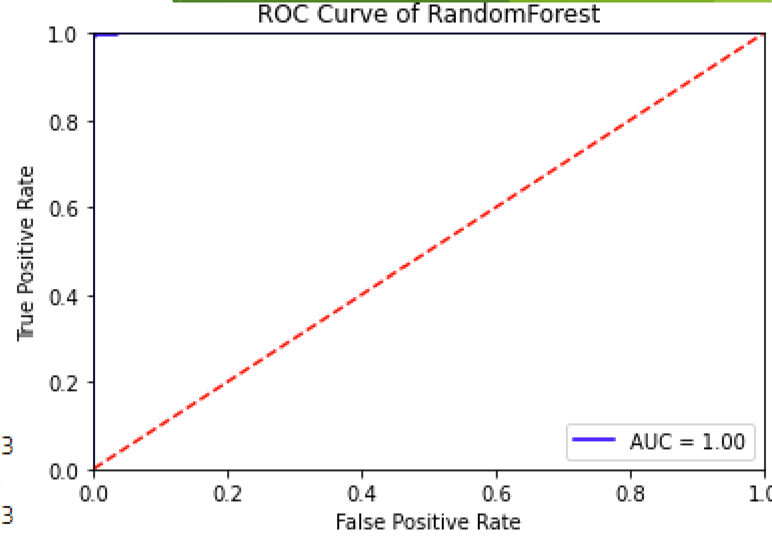


```
Acurácia Média: 0.99 | Desvio: 0.0063
Precisão Média: 0.99 | Desvio: 0.004
Revocação Média: 0.97 | Desvio: 0.023
F1 Score Média: 0.98 | Desvio: 0.011
```

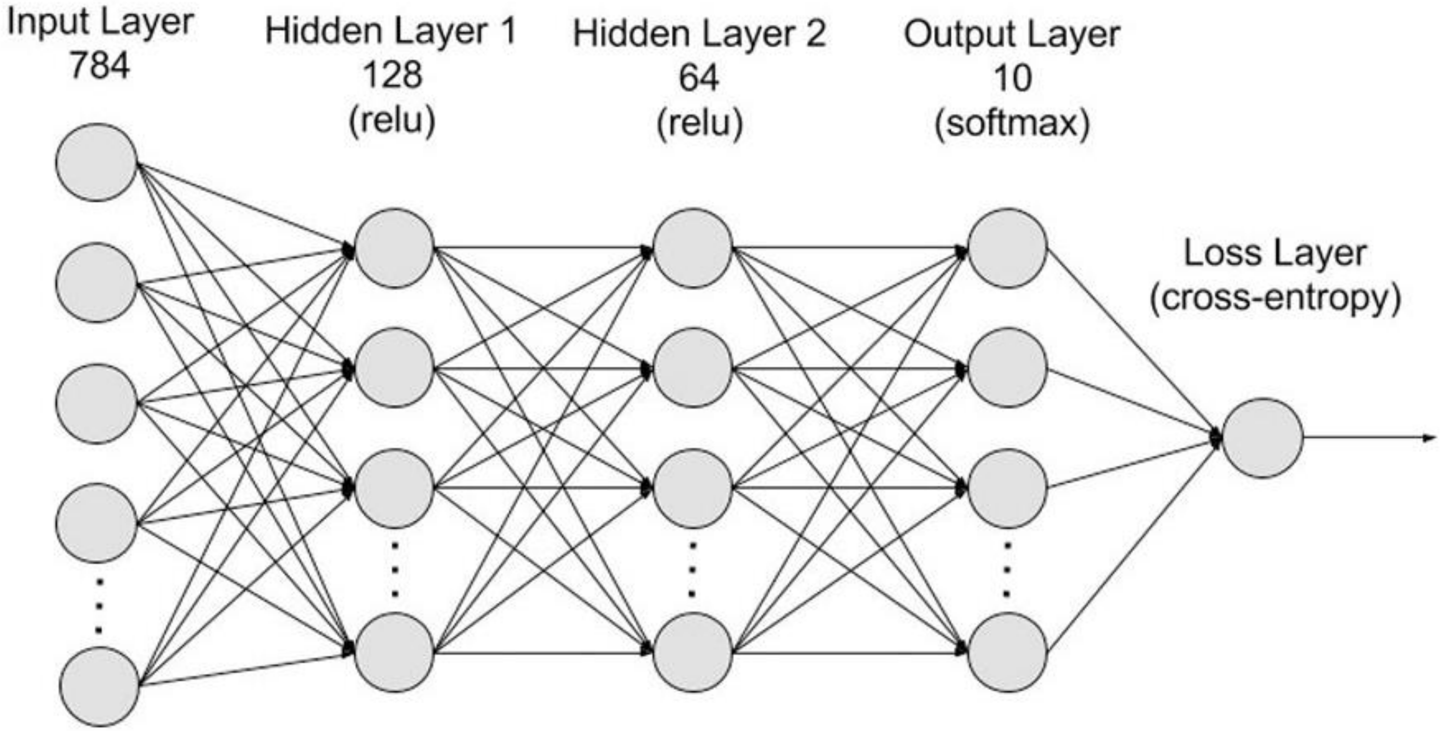


```
Confusion Matrix kfold:
[[5173   17]
 [  55 2125]]
```

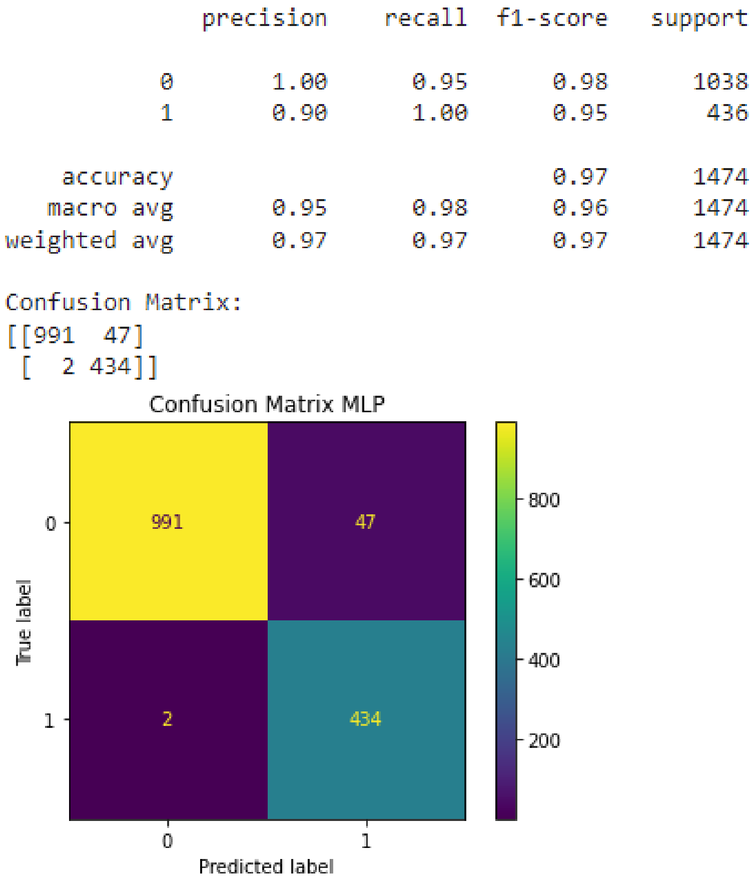




MLP

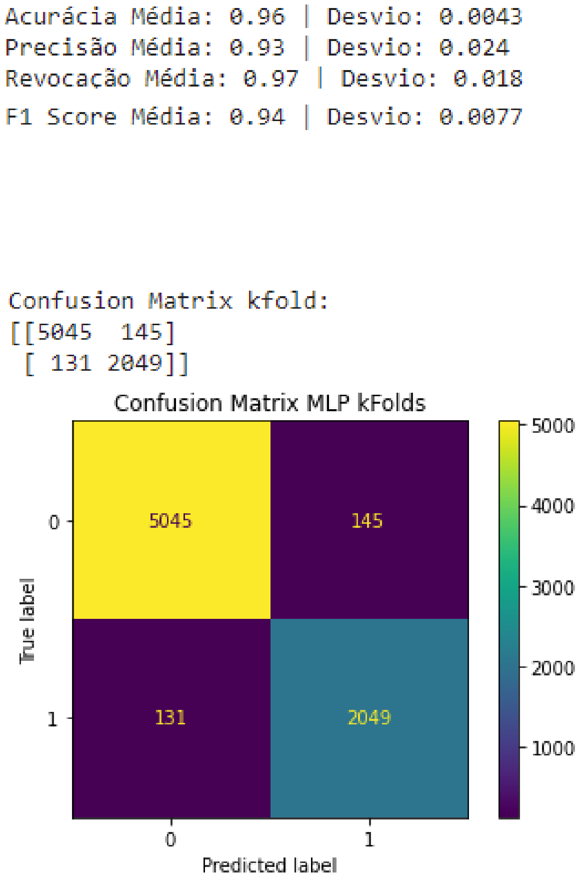

Input Layer
784
Hidden Layer 1
128
(relu)
Hidden Layer 2
64
(relu)
Output Layer
10
(softmax)
Loss Layer
(cross-entropy)

MLP



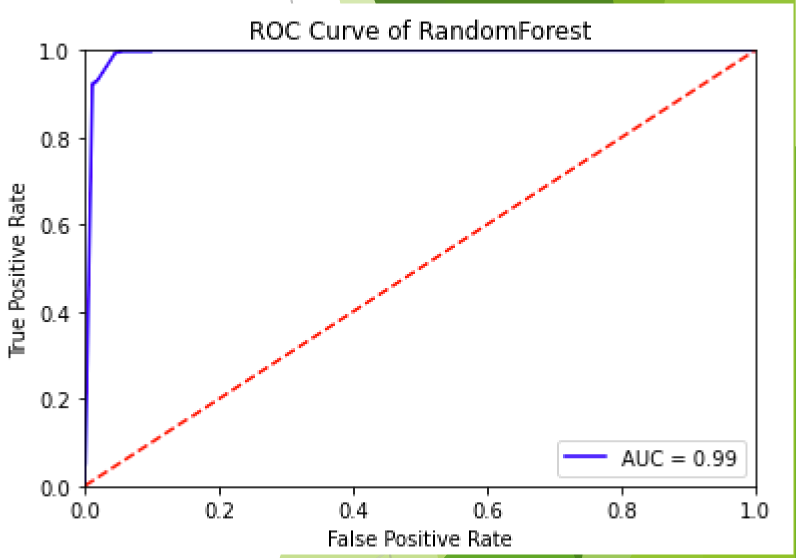
```
              precision    recall  f1-score   support

           0       1.00      0.95      0.98      1038
           1       0.90      1.00      0.95       436

    accuracy                           0.97      1474
   macro avg       0.95      0.98      0.96      1474
weighted avg       0.97      0.97      0.97      1474

Confusion Matrix:
[[991  47]
 [  2 434]]
```

a



```
Acurácia Média: 0.96 | Desvio: 0.0043
Precisão Média: 0.93 | Desvio: 0.024
Revocação Média: 0.97 | Desvio: 0.018
F1 Score Média: 0.94 | Desvio: 0.0077

Confusion Matrix kfold:
[[5045  145]
 [ 131 2049]]
```

b



REPRODUTIBILIDADE

GitHub

https://github.com/diogobortolini/CI1030-Ciencia-de-Dados-para-Seguranca